Transcript for the Summative Assessment presentation submitted as part of the Team presentation for Neural Network Models for Object Recognition.

Provided by: Group 1.

Contents

Slide 1: Title

This is a presentation about our team project on an object recognition model developed using a Neural Network.

Slide 2: Contents

The contents of this presentation are brief introduction about the Neural Network Aim of our project, description of dataset used and Data split, details of data Preprocessing - design of our model,then we talk about overcoming model overfitting problems, and Model evaluation with raining accuracy based on the parameters kernel size, number of kernels, Early Stopping, number of Epochs used and no.of convolution layers. Finally we discuss of findings and provide conclusion

Slide 3: Introduction

Object recognition is one of the important applications of deep learning. With the development of digital cameras and computing power, machines can perceive the world as humans do. With this perception the machines can be programmed to perform tasks such as driving a car, and reading and comprehending hand-written documents. Convolution neural networks help with object detection through feature learning while Artificial neural networks help to classify the detected object.

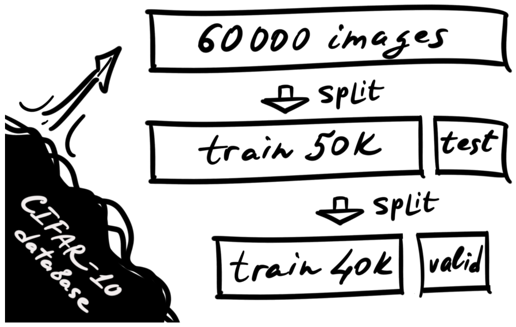
Slide 4: Neural network design

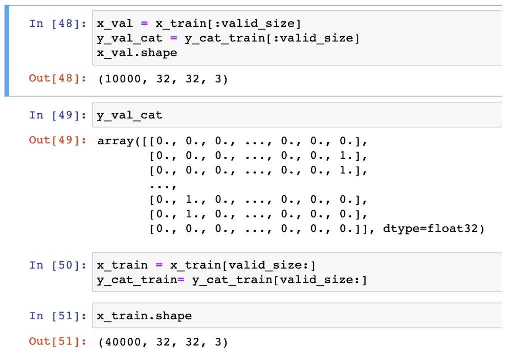
The model performs 2 fundamental steps such as feature learning and classification. CNN provides feature learning with 3 steps: filtering, detection and pooling. In filtering, weighted kernels scan the input image to determine the features in the image, while the activation function ReLU detects the important features and discards unimportant ones. These important features are enhanced by pooling

Slide 5: Aim

Our team aimed to build a neural network for object recognition and to evaluate the built model. Open source CIFAR-10- Object Recognition image dataset, hosted in Kaggle was used in our project which has 60000 colour images of 32 x 32 pixel size, divided into 10 classes, with 6000 images per class. The 10 classes are airplane, automobile, bird, cat, dear, dog, frog, horse, ship and truck. Python libraries such as Keras, TensorFlow, NumPy, pandas, matplotlib were used

Slide 6: Data split





```
In [48]: x_val = x_train[:valid_size]
         y_val_cat = y_cat_train[:valid_size]
         x_val.shape

Out[48]: (10000, 32, 32, 3)

In [49]: y_val_cat

Out[49]: array([[0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 1.],
                [0., 0., 0., ..., 0., 0., 1.],
                ...,
                [0., 1., 0., ..., 0., 0., 0.],
                [0., 1., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)

In [50]: x_train = x_train[valid_size:]
         y_cat_train= y_cat_train[valid_size:]

In [51]: x_train.shape

Out[51]: (40000, 32, 32, 3)
```

Out of the 60000 images in the CIFAR-10 dataset, 50K are training images, and the remaining 10K are test images. However, now we are going to pick a bunch of 10K images from the training pack to create a validation dataset. The validation set is used to validate our model performance during training. The model is simultaneously trained and evaluates the validation set after every epoch. The training loss indicates how well the model fits the training data, while the validation loss indicates how well the model fits new data.

Slide 7: Pre-processing Normalising

Two main steps for the data Preprocessing - normalizing the data and encoding to categories. In simple words for normalizing we should divide each pixel value by 255. The value of every pixel in every image ranges from 0 to 255 for each channel: Red, Green and Blue. Dividing it by 255 we normalize each of them from 0 to 1 range, which is more appropriate for further calculations.

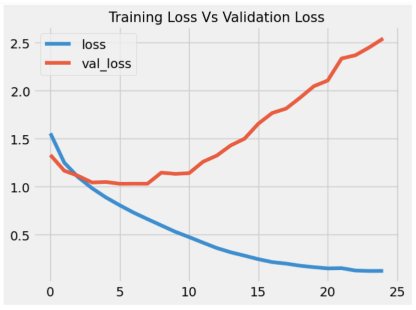
Slide 8: Pre-processing Categorizing

After that we convert a class vector, represented by integers for every class such as "car" or "dog", to a binary class matrix, for example, for the “cat” we have an array of zeros and only index number 4 has the variable of 1, this array encodes our cat, for another class it will be another order of zeros and one, and so on.

Slide 9: Model design

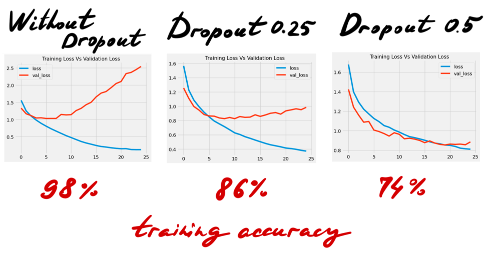
This is a sequentially built model with a stack of convolutional, pooling, flattening, condensing and classifying layers executed sequential order. Initially it has two convolutional layers with 32 filters, that is kernels with a size of 4x4 pixels. The input image is 32x32 pixels with 3 channels for Red, Blue and Green with rectifier linear unit called ReLU activation function performing maximum value pooling. Here ReLU is used as an activation function because it has the advantage that it doesn't activate all the neutrons at the same time. Softmax function is used for the last layer of classification.. categorical_crossentropy: is the loss function used since this is a multi-class classification model that has more number of output labels, which is 10 in our case

Slide 10: Overfitting alarm



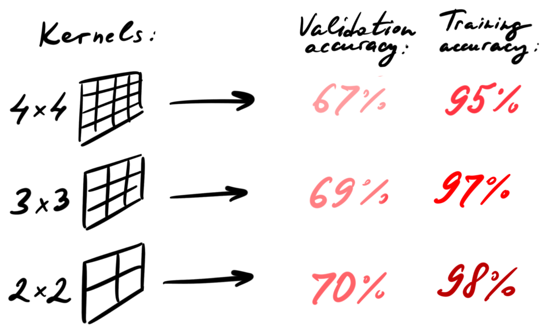
Our model had a very low training loss but a relatively high validation loss – indicating overfitting. Overfitted neural networks are not good at generalizing. (Doon et al., 2018) offer solutions for this case: to reduce capacity or increase data by data augmentation. So, we have two options; the first is to use the Dropout technique, which randomly drops a few units from the network with some probability during training. The second option is to increase the amount of training data by generating modified versions of the samples.

Slide 11: Dropout technique



Let's try to apply Dropout. According to (Srivastava et al., 2014) it prevents overfitting and helps to provide an effective combination of many different neural network architectures. By “dropping” we mean dropping a small unit out of the network, for instance, some nodes from the hidden layer. We can compare the process of dropout to pruning a tree or a decision tree, as well as an evolution in biology, where nature constantly drops out a half of the genes from one parent and another to get a new combination of genes. Here are three graphs for our model training loss and validation loss. First picture is the result without Dropout, the second one is with probability of Dropout 0.25 after every pooling layer in CNN structure, the third one shows the result with Dropout 0.5 after every pooling layer. In the last case we can see that validation loss and training loss go almost parallel. However, the training accuracy of the model is falling down from the first case 98% to 86% and 74% in the second and third respectively.

Slide 12: Model evaluation - kernel size



However we were able to slightly improve the results of our raw model without the Dropout, just playing with the kernel size. After a couple of experiments we realized that both validation and training accuracy are higher in our model with relatively small kernel size. The academic literature says there is no direct relation between the kernel size and the accuracy. Nevertheless when practically we use larger kernels we may start to lose details in some smaller features. In our case the kernel size 2x2 detects little details better, then, for example, 4x4, potentially missing some tiny nuances on our images, which are not very big themselves actually.

Slide 13: Model evaluation - No.of kernels and early stopping

The model accuracy was evaluated by changing the number of kernels used in the convolution layer. Number of kernels 32 and 64 were compared.Also early stopping with patience parameter 2 was used to stop the model when no further improvement is made in terms of accuracy.

This is done to avoid the overfitting and monitoring of the performance of the model.

Slide 14: Epochs used

The model was initially set for 25 epochs. With the use of early stopping patience parameter 2, it was stopped at 12 as there was no further improvement in model accuracy. The optimal number of epochs depends on the complexity of the data, while too many epochs may cause overfitting of data.

Thus, early stopping monitors the performance of every epoch during training.

Slide15: Model evaluation- no.of convolution layers

The model accuracy was tested by adding one more convolution and pooling layer to the model. There was no appreciable change in the training accuracy but the validation accuracy was found closer to training accuracy as seen in this figure.

By progressing of each epoch the errors on the training set should go down along with the validation set. With observations we can see that the validation error starts to go up leaning towards overfitting. So, bringing on early stopping, one should stop training when the validation error has reached to the minimum.

Slide 16: Discussion

Let me summarize a bit, the key findings relates to the kernel size and the quantity of them. The smaller kernel size increases the accuracy for the training model. Double number of kernels provides better accuracy (+1.2%). Another discover goes with struggling with overfitting. Dropout helps to manage overfitting, but decreases further accuracy. Early stopping trick prevents overfitting on the long distance. And one additional Convolutional + ReLU layer increases accuracy.

Slide 17: Conclusion

As a team, we learned the ways of tuning the hyperparameters of the model that help to improve the accuracy of the model. Basically the CIFAR-10 dataset is divided into a training and a test set, but does not come with a validation set by default. We created a validation set, then we trained the model on the training set, and tested on the validation set to see if it is a good fit. In the end the testing set was used to calculate accuracy: about 71%., and we are happy with that. :)

Slide 18: References

The CIFAR-10 dataset. Available at: https://www.cs.toronto.edu/~kriz/cifar.html [Accessed 1 Dec 2022]

Krizhevsky, A. & Hinton, G. (2009) Learning multiple layers of features from tiny images.

Recht, B., Roelofs, R., Schmidt, L. and Shankar, V. (2018) Do cifar-10 classifiers generalize to cifar-10?. *arXiv preprint arXiv:1806.00451*.

Doon, R., Rawat, T.K. and Gautam, S. (2018). Cifar-10 classification using deep convolutional neural network. In 2018 IEEE Punecon (pp. 1-5). IEEE.

Krizhevsky, A. and Hinton, G. (2010) Convolutional deep belief networks on cifar-10. Unpublished manuscript, 40(7), pp.1-9.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014) Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), pp.1929-1958.

Cortes, C., Mohri, M. and Talwalkar, A. (2010) On the impact of kernel approximation on learning accuracy. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 113-120). JMLR Workshop and Conference Proceedings.

Code